Verifying Large SDL-Specifications Using Model Checking

Natalia Sidorova[1] and Martin Steffen[2]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology
Den Dolech 2, P.O.Box 513,
5612 MB Eindhoven, The Netherlands
n.sidorova@tue.nl
[2] Institut für angewandte Mathematik und Informatik
Christian-Albrechts-Universität
Preußerstraße 1–9, D-24105 Kiel, Deutschland
ms@informatik.uni-kiel.de

Abstract. In this paper we propose a methodology for model-checking based verification of large SDL specifications. The methodology is illustrated by a case study of an industrial medium-access protocol for wireless ATM. To cope with the state space explosion, the verification exploits the layered and modular structure of the protocol's SDL specification and proceeds in a bottom-up compositional way. To make a compositional approach feasible in practice, we develop a technique for closing SDL components with a chaotic environment without incurring the state-space penalty of considering all possible combinations of values in the input queues. The compositional arguments are used in combination with abstraction techniques to further reduce the state space of the system. With debugging the system as the prime goal of the verification, we corrected the specification step by step and validated various untimed and time-dependent properties until we built and verified a model of the whole control component of the medium-access protocol. The significance of the case study is in demonstrating that verification tools can handle complex properties of a model as large as shown.

Keywords: SDL model checking; abstraction; compositional, bottom-up verification; verification case study.

1 Introduction

Formal methods, most notably model checking, are increasingly accepted as an important part of the software design process [6]. There is a clear tendency to provide validation facilities in the commercial SDL-design tools such as ObjectGEODE [18] and SDT [22]. Currently, these tools allow validation of SDL specifications by means of exhaustive testing. Due to the high cost of errors in the telecommunication system design, complementary ways of debugging and verification are needed. In this paper, we describe the verification methodology

R. Reed and J. Reed (Eds.): SDL 2001, LNCS 2078, pp. 403–421, 2001.

we applied to a large industrial software product, namely the control layer of the wireless ATM communication protocol *Mascara* [24].

Formal verification of SDL-specifications via model checking [5] is an area of active investigation [3,10,11,8,23] (notably, the last two mentioned citations are developments of the telecommunication industry itself). The "push-button" appeal is responsible for the increasing acceptance of model checking by industry with its promise to allow for fully automatic checking of a program or a system — the model — against a logical specification: typically a formula of some temporal logic. As model checking is based on state-space exploration, the size of a system that can be checked is limited, and it is often held that only relatively small systems can be verified with a model checker.

The limitations of model checking by the system size implies that verification is possible only using abstractions and/or compositional techniques. These techniques allow construction of a verification model whose state space is smaller than the one of the original system. However, providing a formal proof of correctness for each abstraction or composition step is prohibitively costly. If the aim is primarily debugging, performing these steps at a semi-formal level does not cause difficulties, as spotted errors can easily be validated afterwards. The errors can be checked against the concrete model by the designers so that spurious errors can be detected. But if a property holds for the verification model, one cannot claim that the property necessarily holds for the system under consideration as well, although the obtained result argues in favour of correctness of the system design. Therefore, we see that the primary goal of verification is advanced debugging and finding potential errors in its design and thus increasing its reliability, rather than proving the overall correctness of the product.

For the verification of Mascara, we use the Vires tool-set on the SDL specification, automatically translating the SDL-code into the input language of a discrete-time extension of the well-known *Spin* model-checker. As Mascara is too large to be verified by any existing verifier as a whole, we exploit the protocol's layered structure and perform a bottom-up, compositional verification. In a number of cases, the the basis of abstraction of a component are its proved correctness requirements. This abstraction replaces the real component at the next step when a slice at an upper hierarchical level of the protocol is considered for verification. Doing so we were able to reach the point where the whole control entity of Mascara together with a simple abstraction of the rest of the protocol was taken into account.

The rest of the paper is organised as follows: In Sects. 2 and 3 we briefly survey the protocol and the set of design and model check tools we used in the case study. In Sect. 4 we present the methodology and the techniques applied in the verification, and in Sect. 5 we highlight results of the investigation. We conclude in Sect. 6 by evaluating the results and discussing related work.